E0-243 High Performance Computer Architecture

Akash Maji 24212
akashmaji@iisc.ac.in

20 Oct, 2024

We are using the `perf mem` tool to obtain a sampled report of TLB misses occurring at different logical addresses. We then compute the top N large page regions where the number of TLB misses is higher. Then we allocate N large pages to these 2MB regions, and see if any improvement is seen or not. `perf` is a powerful performance analysis tool for Linux that provides various commands to monitor and analyze system and application performance.

We are using a linux system, with necessary tools installed and proper configurations made.

1. Install perf:
   sudo apt-get install linux-tools-common linux-tools-generic linux-tools-`uname -r`
2. Configure kernel settings and huge pages:
   sudo sysctl vm.nr_hugepages=1024
   sudo sysctl kernel.perf_event_paranoid=-1
3. Clean build the files:
   make clean
   make
4. Collect the report:
   sudo perf mem record ./main 24212
   sudo perf mem report > perf.txt

How to collect memory access data?
We run the “perf mem” tool with the ‘record’ option and specify the executable filename with arguments. This will generate a “perf.data” file, which can be read into a human readable text file using the “report” option in the “perf mem” tool. This is shown in step4.

How to analyze collected sampled data?
Once the perf.txt file is available, we can analyze it using a Python program to figure out how many TLB misses are occurring in each of the 2MB regions accessed by the program.
Our Python script (analyse.py) takes in an argument (N) and produces a list of top N addresses in decimal in a text file largepages.txt. Internally, we obtain a sorted list of <2MB Region, TLB Misses> pairs, and then obtain the top N suitable 2MB base regions that we should target to allocate N 2MB large pages. This can be done using mmap() syscall in the modified main program. The Python script will produce a text file of those top N 2MB regions, specifying addresses in decimal. The modified main program will read in these addresses from the text file , and allocate large pages dynamically. We can then rerun the commands in step4, and obtain the new perf.txt report. We can analyze this file to see reductions in TLB misses.

We can additionally use the inbuilt tool “perf stat” to see the improvements in these parameters:

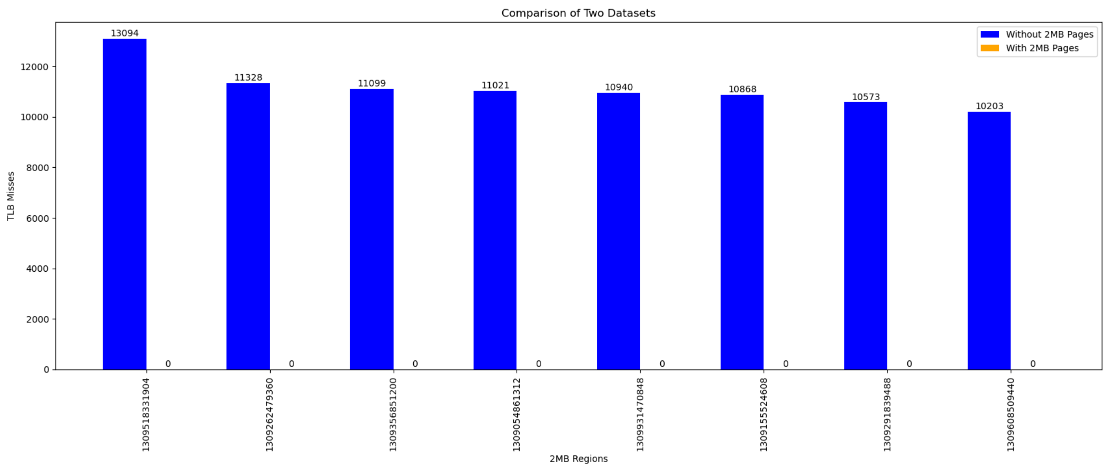
DTLB loads, DTLB load misses, DTLB stores, DTLB store misses

We run this command before and after making changes to main program:

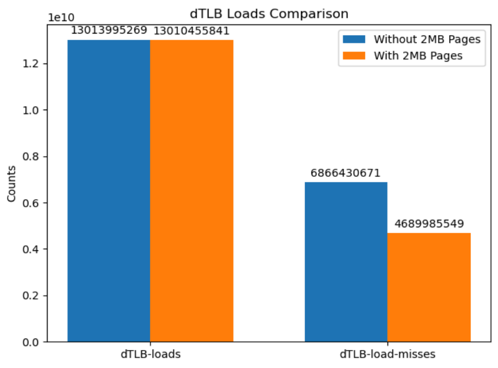
```
make clean
make
sudo perf stat -e dTLB-loads,dTLB-load-misses,dTLB-stores,dTLB-store-misses ./main 24212
```

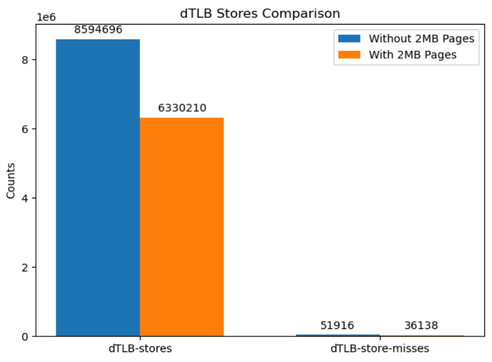
Analysis and Observations:

For N=8, the original main program identified some base regions, and corresponding TLB misses, as per the text report analyzed by script. The modified main program is allocating 2MB pages in these addresses, and corresponding TLB misses. X-axis shows base regions, Y-axis shows TLB misses.
Clearly, we see an improvement in the number of TLB misses occurring in those top N regions.
The number of TLB misses in the report gets reduced to 0 for each of those N regions.
The average speedup is high, indicating a major improvement, as per the new sampled report.



We also ran the “perf stat” tool to see the behavior in dTLB before and after making changes to the main program. We see an improvement of [1.00027, 1.35772] in loads and stores respectively, and improvement of [1.46406, 1.43660] in load-misses and store-misses respectively. The time taken without large pages is **30** sec, and with large pages is **20** sec, with time speed up of **1.5.**

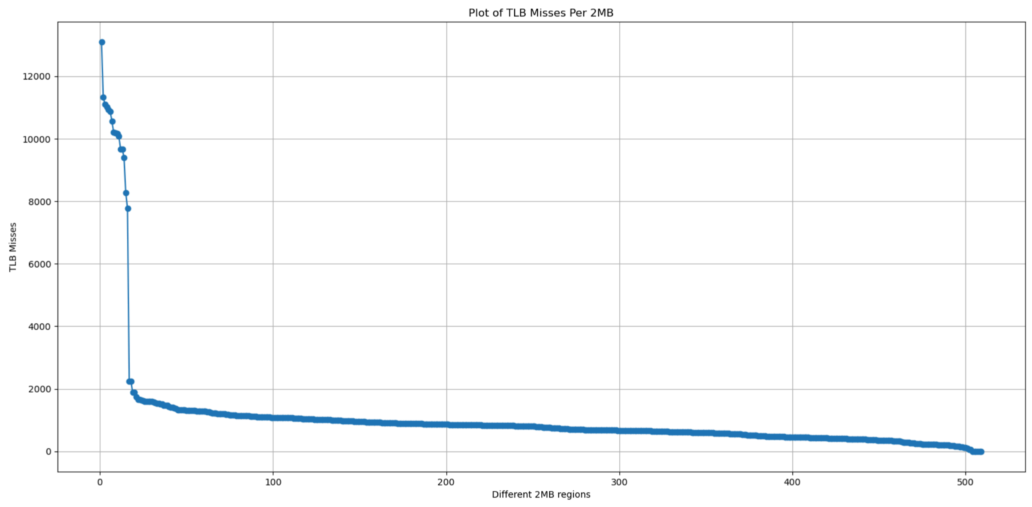




We obtained a graph of TLB misses for each of the at max 512 2MB regions to see how many TLB misses are occurring across different 2 MB regions.

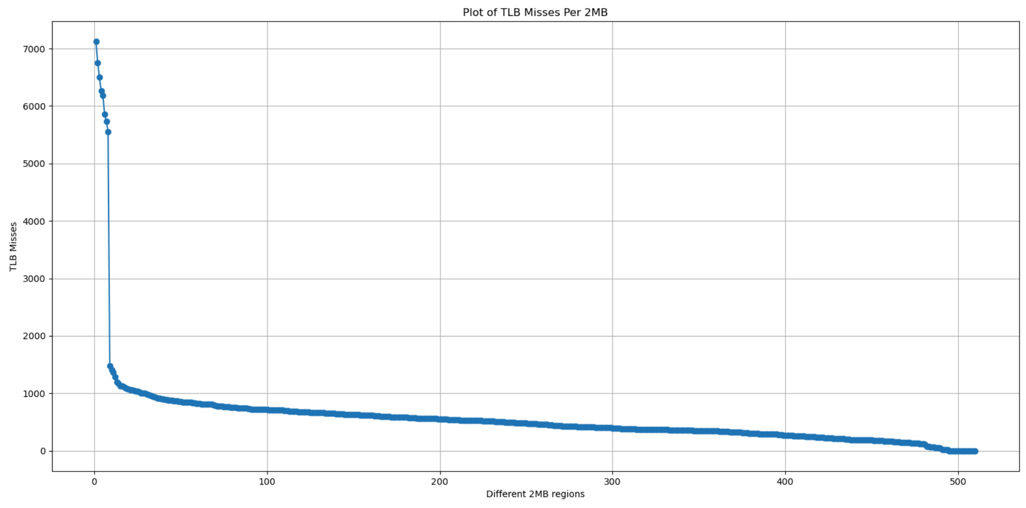
<u>Before running modified main program</u>

Most of the pages incur less than 2000 TLB misses. The total TLB misses is `539774`.



<u>After running modified main program</u>

Most of the pages incur less than 1000 TLB misses. The total misses is `292459`.



The average speedup is **539774 / 292459 = 1.84563.**

References:

- https://www.man7.org/linux/man-pages/man2/mmap.2.html
- https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/9/html/monitoring_and_managing_system_status_and_performance/getting-started-with-perf_monitoring-and-managing-system-status-and-performance#introduction-to-perf_getting-started-with-perf